Master's Thesis

Why Do Masked Neural Language Models Still Need Semantic Knowledge in Question Answering?

Cheong-Woong Kang

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology

2021

Why Do Masked Neural Language Models Still Need Semantic Knowledge in Question Answering?

Cheong-Woong Kang

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology

Why Do Masked Neural Language Models Still Need Semantic Knowledge in Question Answering?

A thesis/dissertation submitted to UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Cheong-Woong Kang

11/23/2020

Approved by

Advisor
Kwang-In Kim

Why Do Masked Neural Language Models Still Need Semantic Knowledge in Question Answering?

Cheong-Woong Kang

This certifies that the thesis/dissertation of Cheong-Woong Kang is approved.

11/23/2020

Signature

Advisor: Kwang-In Kim

Signature

Sungbin Lim: Thesis Committee Member #1

Signature

Jaesik Choi: Thesis Committee Member #2

Abstract

Pre-trained language models have widely been used to solve various natural language processing tasks. Especially, masked neural language models, which are composed of huge neural networks that are trained to restore the masked tokens, have shown outstanding performance in many tasks including text classification and question answering. However, it is challenging to identify what knowledge are trained inside due to the 'black box' nature of deep neural networks with numerous and intermingled parameters. There have been recent studies that try to approximate how much knowledge is learned in masked neural language models. However, a recent study reveals that the models do not precisely understand semantic knowledge while they show superhuman performance. In this work, we empirically verify that questions that require semantic knowledge are still challenging for masked neural language models to solve in question answering. Therefore, we suggest a possible solution that injects semantic knowledge from external repositories into masked neural language models.

Contents

List of Figures

List of Tables

List of Abbreviations

ALBERT A Lite BERT for Self-supervised Learning of Language. 1, 6, 7, 9, 12, 15–17

BERT Bidirectional Encoder Representations from Transformers. 1, 6, 7, 9, 12, 15–17

CNN Convolutional Neural Network. 3, 4

MNLM Masked Neural Language Model. 1, 2, 4–7, 9, 10, 13, 15, 17, 18

NLP Natural Language Processing. 1, 3, 5, 18

QA Question Answering. 1, 2, 7, 9, 10

RNN Recurrent Neural Network. 3, 4

RoBERTa A Robustly Optimized BERT Pretraining Approach. 1, 6, 7, 9, 12, 15–17

SQuAD Stanford Question Answering Dataset. 4, 7, 9–11, 13, 16

I Introduction

One of the long-standing goals in NLP is to teach machines to infer knowledge by understanding language [2]. In NLP, QA is a task to find the correct answer for a given question. QA is widely used as a benchmark to test a machine's ability for natural language understanding and reasoning [3].

Pre-trained language models have shown outstanding performances in various NLP tasks by learning from a huge amount of data. MNLMs are defined as neural language models that are pre-trained to restore the randomly masked words in a sequence of words. Recently, MNLMs, such as BERT [4], ALBERT [5] and RoBERTa [6], have led to a breakthrough in various NLP tasks including QA. However, it is difficult to identify what knowledge contributes to performance improvement and what remains untrained because of the 'black box' nature of deep neural networks with numerous and intermingled parameters.

Recently, there have been active efforts to analyze the inner working mechanisms of NLP models. Recent approaches include behavioral tests, input attribution methods, data attribution methods, probing embeddings, probing attention patterns, and fill-in-blank tests. Behavioral tests diagnose model's behaviors for different input examples of specific types of reasoning skills required [7, 8]. Input attribution methods involve finding the most important parts of an input example [9–11]. Similarly, data attribution methods try to find the most influential examples for the prediction of an example [12, 13]. While attribution methods focus on finding the most important part of inputs or data, probing methods [14–18] focus on exploring linguistic features trained inside models based on embeddings for each layer, such as part-of-speech tagging, named entity recognition, tense analysis, parsing and chunking. A common way for probing linguistic features is to verify the existence of linguistic knowledge by training a simple neural network for each linguistic feature of interest. There have been recent studies that probe attention patterns without additional training on linguistic features [19, 20]. However, a recent study argues that attention cannot be used as an explanation [21]. A fill-in-blank style test is a more intuitive analysis method that explores relational knowledge learned in deep neural language models. Specifically, [22] observes that BERT has learned relational knowledge, as competitive as knowledge bases. However, [23] finds that BERT has learned shallow pattern matching rather than the recall of learned factual knowledge.

Semantic knowledge is known to be an important factor for natural language understanding and inference in QA tasks. As studied in [23], MNLMs rely on spurious statistical patterns instead of understanding semantic knowledge while they show outstanding performance in various tasks. Specifically, they find that MNLMs do not often distinguish two opposite relations. Extend this finding to QA tasks, we begin with two hypotheses:

1. Questions that require semantic knowledge are still difficult for MNLMs to solve while they show outstanding performance in QA.

2. MNLMs can be supplemented by integrating semantic knowledge from external knowledge repositories.

To test our first hypothesis, we analyze the effect of lexical overlaps between a question and a context on the difficulty of a question. If a question and a context have less lexical overlaps, they would have more semantic variations. On the other hand, if a question and a context have more lexical overlaps, they would have less semantic variations. Since less lexical overlaps do not always guarantee more semantic variations, we also try with a different but more detailed analysis method to categorize questions into six types based on required reasoning skills to solve the questions. As a result, we find that questions that require semantic knowledge, specifically commonsense knowledge, are still challenging for MNLMs to solve.

For our second hypothesis, we suggest a way to utilize semantic knowledge from external knowledge sources into MNLMs. Specifically, we use ConceptNet [24] as an external knowledge repository for our experiments. To incorporate external semantic knowledge, we propose an automatic approach as well as a manual approach. The experimental results demonstrate that incorrectly predicted questions can be correctly answered by MNLMs with the help of external knowledge.

The main contributions in this thesis are as follows:

- Questions that require semantic knowledge, especially commonsense knowledge, are still difficult for MNLMs to solve while they show outstanding performance in QA.
- We propose a way to dynamically integrate external semantic knowledge to supplement MNLMs.

II Background

2.1 Text Representation

Language Modeling

A language model is defined as a probability distribution over word sequences. Given a sequence of m words $w_1, w_2, \ldots, w_m$, it assigns a probability $P(w_1, w_2, \ldots, w_m)$. For example, a sequence of words "Today is Monday" would have a higher probability than "Today Monday is". [25] trains a language model by predicting the next word of a sequence of words with a neural network. A probability distribution of the next word $x^{(t+1)}$ given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$ is defined as $P(x^{(t+1)}|x^{(t)}, x^{(t-1)}..., x^{(1)})$, where $x^{(t+1)}$ can be any word in the vocabulary $V = w_1, w_2, \ldots, w_{|V|}$. For example, given a sequence of words "I want a cup of", the next word would be 'water' or 'milk', rather than 'cake' or 'laptop'. Language modeling is one of the most essential parts of NLP since it is used to learn text representations in various applications, such as machine translation, question answering, sentiment analysis, and text summarization.

Word Embeddings

There have been various approaches [26–28] to learn a distributed representation for words from large text corpora. Although they are widely used in various NLP applications, they suffer from distinguishing multiple senses of a word (polysemy). Specifically, a word 'bank' in two different contexts "open a bank account" and "on the river bank" has different senses. Since they assign the same representation for a word 'bank' regardless of which sense of the word is used in different contexts, they often fail to encode multiple senses in a single representation.

Contextualized Embeddings

Recent studies [4, 29] use context-dependent representations to solve the limitations of existing word embeddings. Instead of using a single representation for each word, they use different representations for different contexts in which the word appears. They show that using context-dependent representations can significantly improve existing models in various NLP benchmarks.

2.2 Transformer

[1] proposes Transformer, a neural network architecture that is merely based on attention mechanisms, to address the limitations of existing architectures based on RNN and CNN. RNN has a computational burden especially for longer sequences of words because of its sequential nature. There have been numerous efforts [30–32] to reduce the computational burden by using CNN. However, both RNN and CNN suffer from learning dependencies between two distant words. Attention mechanisms [33] are proposed to model dependencies regardless of a distance between words. However, existing methods [34–36] use attention mechanisms in conjunction

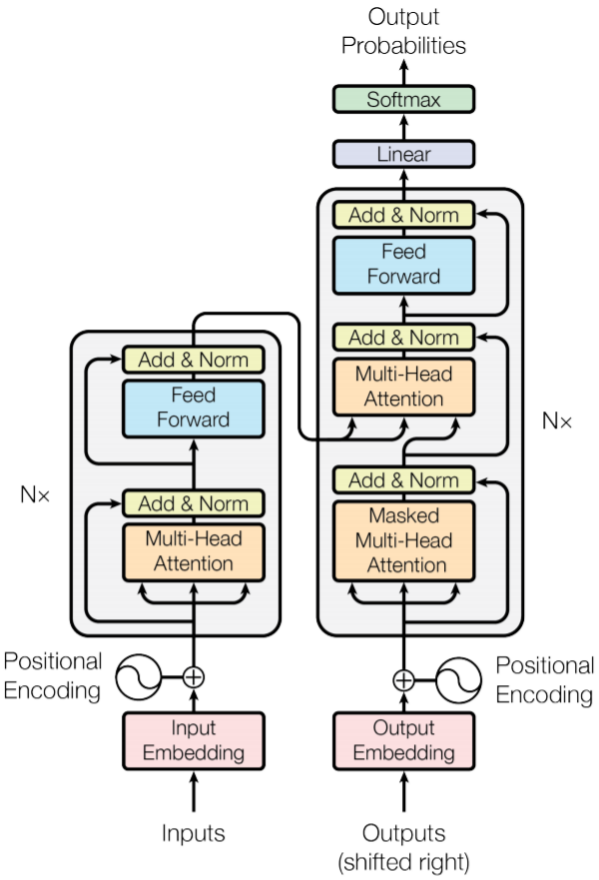

Figure 1: The overall architecture of Transformer. Reprinted from Vaswani et al. NIPS 2017:5998-6008 [1].

with RNN so the computational burdens still remain. On the other hand, Transformer is solely based on attention mechanisms so the limitations of RNN and CNN are resolved. The overall architecture of Transformer is shown in Figure 1. Each component of Transformer is described below.

Input Embedding

The input sequence of words are mapped to the sequence of vectors, using learned embeddings. Any embedding algorithms can be used here.

Positional Encoding

Since Transformer is solely based on attention mechanisms without RNN and CNN, positional information is needed. Therefore, positional encodings are added to the input embeddings. The positional encodings and input embeddings can simply be summed because they have the same dimension. Sine and cosine functions are used in [1] while there are many choices of positional encodings [30].

Multi-head Attention

Multiple attention functions rather than a single attention function are used in each layer. Specifically, word embeddings are linearly projected to h vectors with the size of $\frac{d}{h}$, where h is the number of heads and d is the dimension of word embeddings. The outputs are concatenated after performing attention functions. Multi-head attention allows the model to use information from multiple representation subspaces.

Scaled Dot-product Attention

Scaled dot-product attention is used as an attention function in [1]. Query, key and value weight matrices are trained to transform the input embeddings to query, key and value vectors. To compute attention weights, the dot product of query and key vectors are scaled (divided by d_k) and a softmax function is applied. Then, the value vectors are multiplied by attention weights to yield the outputs.

Feed Forward Networks

Followed by multi-head attentions, feed forward networks are applied in a position-wise manner (the networks are identically applied regardless of positions). Feed forward networks consist of two linear layers with a activation function in between.

Residual & Layer Normalization

Residual connections and layer normalization are applied after every attention and feed forward block. The input and output vectors of each block are summed, and normalized in a layer-wise manner.

Linear & Softmax

Linear layers and a softmax function are applied to convert the decoder output to predicted next word probabilities.

2.3 Masked Neural Language Models

Recently, MNLMs have been used to solve various NLP tasks, showing superhuman performances [4–6]. MNLMs can be fine-tuned to various tasks after pre-training with two objectives: 1) masked language model and 2) next sentence prediction. Different from traditional neural language models [25] that are trained with next word prediction, MNLMs use a masked language model objective, which restores the randomly masked tokens in a sequence of tokens. They additionally use a next sentence prediction objective, which predicts whether a sentence B is likely to appear after a sentence A. In the following, we briefly introduce MNLMs that are

Table 1: Structures of MNLMs.

Models	Number of Parameters	Number of Layers	Hidden Size	Data Size
BERT_{base}	108M	12	768	16GB
BERT_{large}	334M	24	1024	16GB
ALBERT1_{base}	12M	12	768	16GB
ALBERT1_{large}	18M	24	1024	16GB
ALBERT1_{xlarge}	60M	24	2048	16GB
ALBERT2_{base}	12M	12	768	160GB
ALBERT2_{large}	18M	24	1024	160GB
ALBERT2_{xlarge}	60M	24	2048	160GB
RoBERTa_{base}	108M	12	768	160GB
RoBERTa_{large}	334M	24	1024	160GB

used in our experiments. Detailed information, such as the number of parameters and the size of pre-training data, is described in Table 1.

BERT

BERT [4] is composed of the transformer encoder layers [1]. The model comprises a stack of L transformer encoder layers. Each layer is composed of H self-attention heads and hidden states of D dimensions. The input is represented as a pair of two text sequences $A_1, ..., A_N$ and $B_1, ..., B_M$, where each word token is mapped to its corresponding WordPiece embeddings [37]. Additionally, special token '[CLS]' and '[SEP]', or 'classification token' and 'seperator token' are used for the final input representation:

$$[CLS], A_1, ..., A_N, [SEP], B_1, ..., B_M, [SEP]$$

BERT is pre-trained with about 16GB of text corpora, which is composed of English Wikipedia and Book Corpus [38].

RoBERTa

RoBERTa, which is a variant of BERT, has the same structure with BERT. However, there are several changes to improve the BERT. First, RoBERTa dynamically masks tokens during training while BERT has fixed masked tokens. Second, RoBERTa is trained with a single sequence of tokens instead of using a next sentence prediction objective. Next, RoBERTa uses a byte pair encoding [39] algorithm instead of the WordPiece for tokenization. Lastly, RoBERTa is pre-trained with much larger data. Especially, the data is about 160GB, which is composed of CommonCrawl News dataset [40], Open WebText corpus [41], and STORIES corpus [42] in addition to the pre-training data of BERT.

ALBERT

ALBERT is another variant of BERT, using the same structure with several differences. First, ALBERT uses a decreased number of word embedding dimensions and shares parameters to reduce the complexity. Additionally, ALBERT is pre-trained to predict sentence order instead of the next sentence prediction objective. ALBERT has two versions with varying data sizes. ALBERT1 uses 16GB of data, which is the same as BERT, while ALBERT2 uses 160GB of data, which is the same as RoBERTa.

Semantic Knowledge of MNLMs

Semantic knowledge is key to natural language understanding [2]. There have been several studies that investigate whether MNLMs can recall the semantic knowledge that is part of their training data. Specifically, [22] introduces language model analysis, which is a way to test MNLMs by querying semantic knowledge with a masked word. For example, 'birds can fly' can be converted into 'birds can _____' to test if the model can recall the fact that birds can fly. [22] shows that MNLMs perform better than automatically extracted knowledge bases in language model analysis. However, [23] reveals that MNLMs do not precisely understand semantic knowledge. Especially, they do not even know the meaning of negation, which can be easily captured by humans, in many cases. Therefore, we hypothesize that questions that require semantic knowledge would be still difficult for MNLMs to solve despite their outstanding performance in QA tasks.

2.4 Question Answering

QA is one of widely used benchmarks to test ability of machines to understand natural language. There are several different types of QA tasks but we focus on a specific type of QA task called extractive QA. The task is to extract the answer from the context, given a question and an associated context, where the answer appears as it is in the context.

For experiments, we use SQuAD 2.0 dataset. The dataset consists of *has answer* and *no answer* questions. A *has answer* question can be answered directly or indirectly after reading the context. On the other hand, a *no answer* question cannot be answered even after reading the context.

2.5 ConceptNet

Since the scope of semantic knowledge is too broad, we focus on knowledge from a specific semantic knowledge graph, called ConceptNet [24]. ConceptNet is a semantic knowledge graph extracted from open mind commonsense dataset [43], designed to help machines understand semantic knowledge shared by humans. It has been widely utilized as a knowledge repository in existing studies [44–46]. In ConceptNet, semantic knowledge is represented as a graph, where each node represents a word or an entity, each edge represents a relation between two entities.

We can also think of it as a set of triples, composed of subject, relation, and object (e.g. 'birds', 'CapableOf', 'fly'). For experiments, we use a 5.6.0 version of ConceptNet.

III Do MNLMs Understand Semantic Knowledge in QA?

Table 2: Examples of easy and hard questions on the *has answer* questions.

Similarity*	Question	Answer	Context
> 0.6	In what year did Savery patent his steam pump?	1698	... In 1698 Thomas Savery patented a steam pump that used steam in direct contact with the water being pumped. ...
< 0.1	Which country was the last to receive the disease?	northwestern Russia	From Italy, the disease spread northwest across Europe, ... Finally it spread to northwestern Russia in 1351. ...

* Cosine similarity between TF-IDF term weighted uni-gram vectors of the question and the context

Although MNLMs have shown outstanding performance in various tasks, they have incomplete semantic knowledge [23]. Extending this to QA, we try to find whether the models understand semantic knowledge or not. We assume that a similarity between question and context is related to the amount of required semantic knowledge. For example, if a question and a context are similar, or have many overlapping words, they are less likely to require semantic knowledge. On the other hand, if a question and context are not similar, they are more likely to require semantic knowledge. The examples of questions with different similarity scores are shown in Table 2. We observe that the first question is similar to context, therefore the clues are explicitly shown in the context. On the other hand, the second question is not similar with the context, thus requires additional clues that are not explicitly provided. For example, to solve the question, we have to infer that Russia is a country, as well as that 'spread the disease' can be paraphrased into 'receive the disease'. Therefore, questions with low similarity are more likely to require implicit clues, such as semantic knowledge, affecting the difficulty of the questions. In the following, we analyze a correlation between similarity and difficulty of the question. In addition, we try with a different but more detailed analysis method, which is to manually categorize the questions into six types according to the reasoning skills required to solve the questions. Then, we analyze what types of questions are difficult to solve so that we can test whether semantic knowledge required questions are challenging or not. For analysis, we use largest models of each type of MNLMs trained on the SQuAD 2.0 [47], a QA dataset: BERT$_{large}$, ALBERT1$_{xlarge}$, ALBERT2$_{xlarge}$ and RoBERTa$_{large}$. The analyses are conducted with the development set since we cannot access the test set. The detailed information is explained in the following.

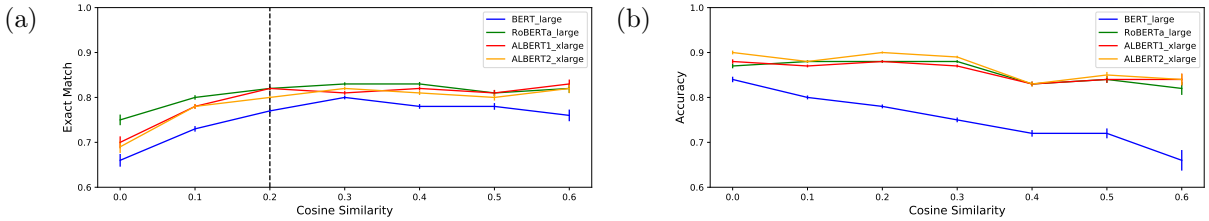

Figure 2: Results on the correlation between similarity and difficulty of questions.

3.1 Correlation between Similarity and Difficulty of Questions

In this section, we analyze how similarity affects difficulty of questions. Here, similarity means similarity between a question and a context. Specifically, we use a cosine similarity between a question and a context, represented as uni-gram bag-of-words vectors normalized by term frequency and inverse document frequency [48]. As a measure of difficulty, we use performance of the MNLM-based QA models. Specifically, we use an exact match score and an accuracy for *has answer* and *no answer* questions, respectively. The results are shown in Figure 2, where (a) shows results of the *has answer* questions and (b) shows results of the *no answer* questions. X-axis denotes the cosine similarity of a context and a question. Y-axis indicates the performance of the models: an exact match score and an accuracy, respectively. The questions whose similarity score is higher than 0.6 are ignored since there are only a few (less than 1%). The error bars in the graphs are standard deviations. We see that *has answer* questions become more difficult as their similarity is higher while *no answer* questions have the opposite tendency. Especially, among *has answer* questions, the performance drops sharply when the similarity becomes lower than 0.2. The questions whose similarity is below 0.2 account for only 20% of all of the *has ans* questions. We suspect that the high performance of MNLMs is due to the large portion of relatively easy questions.

3.2 Categorizing Questions based on Reasoning Types

Here, we test if questions that require semantic knowledge are difficult to MNLMs in QA by classifying the questions based on reasoning types. We focus on *has answer* questions whose similarity is below 0.2 because we are interested in which types of questions are difficult. We categorize the questions into the six types based on reasoning types in SQuAD [49], as shown in Table 3. Then, we analyze the correlation between these types and the performance of the models. We analyze the proportions of questions in each type comparing questions that are correctly predicted by the models and incorrectly predicted questions. The analysis results are shown in Table 4. We observe differences in the proportion of the questions in each type between the correctly and incorrectly predicted questions. Among the correctly predicted questions, *no semantic variation* questions account for more than 50% across the models. This implies that the questions that do not require semantic knowledge are relatively easy to solve. Among the

Table 3: Examples and descriptions for each type of the *has answer* questions in SQuAD 2.0.

Question Types	Description	Example
Synonymy	There is a clear correspondence between question and context.	**Question**: Which entity is the **secondary** legislative body? **Context**: ... The **second main** legislative body is the Council, which is composed of different ministers of the member states. ...
Commonsense knowledge	Commonsense knowledge is required to solve the question.	**Question**: Where is the **Asian** influence strongest in Victoria? **Context**: ... Many **Chinese** miners worked in Victoria, and their legacy is particularly strong in Bendigo and its environs. ...
No semantic variation	There is no semantic variation such as synonymy or commonsense knowledge.	**Question**: Who are the **un-elected subordinates of member state governments**? **Context**: ... This means Commissioners are, through the appointment process, the **unelected subordinates of member state governments**. ...
Multi-sentence reasoning	Hints for solving questions are shattered in multiple sentences.	**Question**: Why did **France** choose to give up continental lands? **Context**: ... **France** chose to cede the former, ... **They** viewed the economic value of the Caribbean islands' sugar cane ...
Typo	There exist typing errors in the question or context.	**Question**: What kind of measurements define **accelerlations**? **Context**... **Accelerations** can be defined through kinematic measurements. ...
Others	The labeled answer is incorrect.	**Question**: Who **won the battle** of Lake George? **Context**: ... The **battle ended inconclusively**, with both sides withdrawing from the field. ...

Table 4: The analysis results of question type categorization.

Model	Status*	Question type					
		Semantic variation		Multiple sentence reasoning	No semantic variation	Typo	Others
		Synonymy	Commonsense Knowledge				
BERT_{large}	Correct	27.29	22.54	11.82	**54.14**	1.22	1.10
BERT_{large}	Incorrect	30.71	**49.46**	19.84	27.45	1.63	2.99
ALBERT1_{xlarge}	Correct	28.06	24.40	12.46	**51.52**	1.15	1.15
ALBERT1_{xlarge}	Incorrect	28.93	**48.11**	19.18	31.13	1.89	3.14
ALBERT2_{xlarge}	Correct	28.22	24.41	12.56	**51.80**	1.13	0.93
ALBERT2_{xlarge}	Incorrect	28.48	**49.34**	19.21	29.14	1.99	3.97
RoBERTa_{large}	Correct	28.81	25.12	12.36	**50.55**	1.30	1.10
RoBERTa_{large}	Incorrect	26.30	**49.63**	20.74	31.11	1.48	3.70
Overall†	Correct	27.29	17.86	10.97	**57.81**	1.27	0.84
Overall	Incorrect	25.69	**55.96**	22.02	24.77	1.83	4.59
Total proportion		28.28	30.32	14.14	46.43	1.34	1.65

The categories can be tagged with duplicates except for semantic variation and no semantic variation.

* Correct: Questions correctly predicted by the model, Incorrect: Questions incorrectly predicted by the model

† Questions succeed or failed to predict by all experimental models commonly

incorrectly predicted questions, *commonsense knowledge* questions account for about 50% across the models. Therefore, the questions that require commonsense knowledge, which is a kind of semantic knowledge, are still the most difficult for the models to solve.

IV Integrating Semantic Knowledge from External Knowledge Repositories

Table 5: Examples of manual external commonsense knowledge integration.

Required Knowledge*	Question	Context
uv Synonym ultraviolet radiation	_____ Helps the biospher from **UV**, **which is the same as ultraviolet radiation**.	... the high-altitude ozone layer helps protect the biosphere from **ultraviolet radiation**, ...
rare Antonym frequent	How **frequent** is snow in the Southwest of the state?	... But snow is very **rare**, **which is the opposite of frequent**, in the Southwest of the state, ...
punishment RelatedTo sentence	Why would one want to give more **punishment**, **which is related to sentence**?	... the judge increased her **sentence** from 40 to 60 days. ...

* The blue words indicate the subject term of the triple
The red words indicate the relation or the relation's relative pronoun template of the triple
The turquoise words indicate the object term of the triple

We found that MNLMs have limitations in solving questions that require commonsense knowledge. Therefore, we suggest a possible solution that injects semantic knowledge, especially commonsense knowledge, from external repositories into the models. First, we suggest a manual knowledge integration method, where a human manually integrates external knowledge into the models. However, it is not always possible for humans to manually incorporate knowledge. Therefore, we suggest an automatic approach as well. For experiments, we use ConceptNet [24] as an external knowledge source. The experiments on knowledge integration are described in the following.

4.1 Manual Knowledge Integration

We first manually integrate external knowledge. Specifically, we inject knowledge by augmenting the text of a question or a context without additional training or changing the model structure. We illustrate how the required knowledge is incorporated into the text in Table 5. First, we find a subject word of a required knowledge triple in the question and context. Then, we augment the word with the relation and object of the triple. Here, the relation is converted into a natural language form using the templates. The templates used in the experiments are shown in Table 6.

We test our manual knowledge integration method on questions in SQuAD 2.0 that are

Table 6: The templates for 37 relations in ConceptNet.

Relation	Template
RelatedTo	[[SUBJ]] is related to [[OBJ]] .
HasContext	[[SUBJ]] is used in the context of [[OBJ]] .
IsA	[[SUBJ]] is a [[OBJ]] .
DerivedFrom	[[SUBJ]] is derived from [[OBJ]] .
Synonym	[[SUBJ]] is the same as [[OBJ]] .
FormOf	[[SUBJ]] is the form of [[OBJ]] .
SimilarTo	[[SUBJ]] is similar to [[OBJ]] .
EtymologicallyRelatedTo	[[SUBJ]] is etymologically related to [[OBJ]] .
AtLocation	[[SUBJ]] can be found at [[OBJ]] .
MannerOf	[[SUBJ]] is a way to [[OBJ]] .
Antonym	[[SUBJ]] is the opposite of [[OBJ]] .
HasProperty	[[SUBJ]] can be [[OBJ]] .
PartOf	[[SUBJ]] is part of [[OBJ]] .
UsedFor	[[SUBJ]] may be used for [[OBJ]] .
DistinctFrom	[[SUBJ]] is not [[OBJ]] .
HasPrerequisite	[[SUBJ]] requires [[OBJ]] .
HasSubevent	[[SUBJ]] has subevent, [[OBJ]] .
Causes	[[SUBJ]] causes [[OBJ]] .
HasA	[[SUBJ]] contains [[OBJ]] .
InstanceOf	[[SUBJ]] is an instance of [[OBJ]] .
CapableOf	[[SUBJ]] can [[OBJ]] .
ReceivesAction	[[SUBJ]] can be [[OBJ]] .
MotivatedByGoal	[[SUBJ]] can be motivated by [[OBJ]] .
CausesDesire	[[SUBJ]] would make you want to [[OBJ]] .
MadeOf	[[SUBJ]] can be made of [[OBJ]] .
HasLastSubevent	[[SUBJ]] has the last subevent [[OBJ]] .
Entails	[[SUBJ]] entails [[OBJ]] .
HasFirstSubevent	[[SUBJ]] has the first subevent [[OBJ]] .
Desires	[[SUBJ]] wants [[OBJ]] .
NotHasProperty	[[SUBJ]] can not be [[OBJ]] .
CreatedBy	[[SUBJ]] is created by [[OBJ]] .
NotDesires	[[SUBJ]] does not want [[OBJ]] .
DefinedAs	[[SUBJ]] can be defined as [[OBJ]] .
NotCapableOf	[[SUBJ]] can not [[OBJ]] .
LocatedNear	[[SUBJ]] is typically near [[OBJ]] .
EtymologicallyDerivedFrom	[[SUBJ]] is etymologically derived from [[OBJ]] .
SymbolOf	[[SUBJ]] is an symbol of [[OBJ]] .

Table 7: Experimental results of the manual knowledge integration test.

Model	Number of examples	Original text		Knowledge integrated text	
		EM*	F1†	EM	F1
BERT_{large}	78	0.00	28.08	8.97	35.22
ALBERT2_{xlarge}	69	0.00	31.86	13.04	44.81
RoBERTa_{large}	64	0.00	32.94	14.06	41.49

* F1 score

† Exact match score

incorrectly predicted by each model, where the exact match score is zero. Note that we focus on *has ans* questions whose similarity score is below 0.2. Additionally, we find that 244 questions out of 684 *semantic variation* type questions have clues in ConceptNet. Then, we use incorrectly predicted questions among 244 questions. Then, we test our method in exact match score and F1 score. F1 score is a harmonic mean of recall and precision, as shown below.

$$\text{Precision} = \frac{\#\text{ of words in predicted answer matched with words in the ground truth}}{\#\text{ of words in predicted answer}} \tag{1}$$

$$\text{Recall} = \frac{\#\text{ of words in predicted answer matched with words in the ground truth}}{\#\text{ of words in the ground truth}} \tag{2}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

The results of manual knowledge integration method are shown in Table 7. Here, we did not test ALBERT1 due to the engineering issues. We see that some of them are correctly predicted across the models after applying our method. This implies that using external knowledge can be a possible direction for overcoming the limitations of existing MNLMs in inferring semantic knowledge, especially commonsense knowledge.

4.2 Automatic Knowledge Integration

Since it is not efficient for humans to identify and integrate required knowledge, we suggest an automatic integration method as well. The overall architecture of the automatic integration model is shown in Figure 3. First, the input text is encoded using MNLMs into the contextualized text representations. At the same time, the input text is passed to external knowledge repositories, such as ConceptNet, to extract candidate knowledge triples. Specifically, we extract candidate triples if either a subject or an object of a triple appears in the input text. After extracting candidate triples, we convert them into vectors using the word embeddings of MNLMs. Since each triple is represented as multiple vectors, we use attention pooling so that each triple can be represented as a single vector. Then, we align commonsense knowledge into

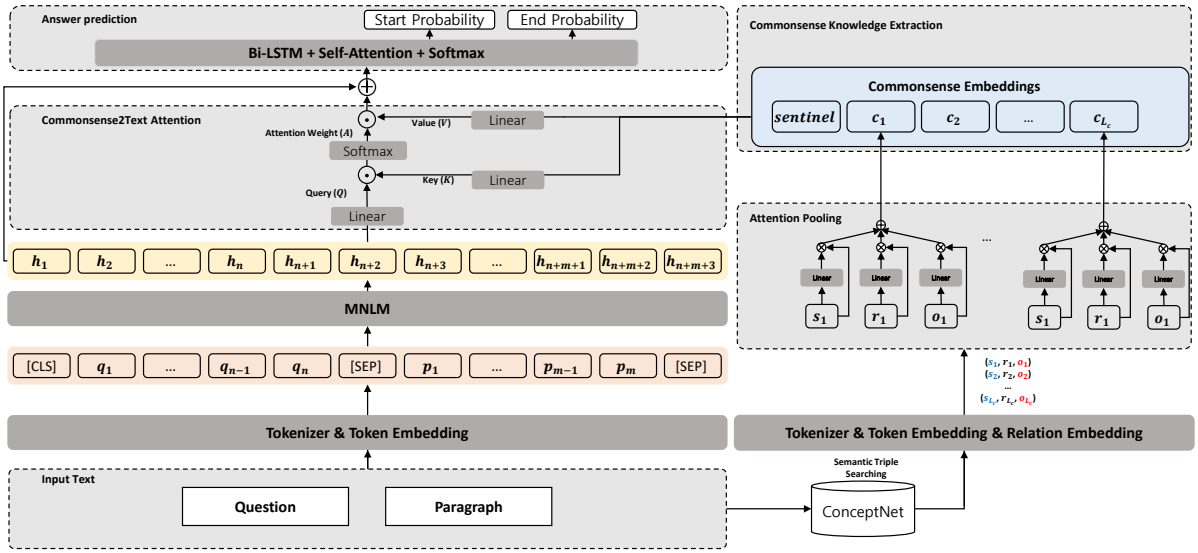

Figure 3: The overall architecture of the automatic knowledge integration model.

Table 8: Experimental results of the automatic knowledge integration test on all of the *has ans* questions.

Model	F1	
	Baseline	Knowledge integrated model
BERT_{base}	78.39	**80.93**
BERT_{large}	82.86	**83.70**
ALBERT2_{base}	80.30	**80.60**
ALBERT2_{large}	84.18	**84.42**
ALBERT2_{xlarge}	86.78	**87.20**
RoBERTa_{base}	80.89	**81.85**
RoBERTa_{large}	88.07	**88.36**

the input text using the attention mechanism [33]. This can be thought as learning to selectively utilize knowledge among candidate knowledge triples. Here, a sentinel vector is used to indicate the cases where none of the candidates are necessary to solve the question. The commonsense aligned text representation is passed to a bi-directional layer and a self-attention layer, respectively. Finally, a softmax function is applied to find the start and end positions of the answer. We first tested our automatic knowledge integration method on all of the *has ans* questions in the SQuAD 2.0 dataset. The experimental results are shown in Table 8. We observe that the performance has improved consistently across the models. In addition, as shown in Table 9, we tested on commonsense required questions among the *has ans* questions that we labeled to see if our method really helps the models infer commonsense knowledge. Specifically, we test on the 386 commonsense required questions. The results show the consistent performance improvement

Table 9: Experimental results of the automatic knowledge integration test on commonsense required questions among the *has ans* questions.

Model	F1	
	Baseline	Knowledge integrated model
BERT$_{base}$	52.47	**57.68**
BERT$_{large}$	61.23	**61.28**
ALBERT2$_{base}$	57.72	**58.26**
ALBERT2$_{large}$	62.59	**63.44**
ALBERT2$_{xlarge}$	69.46	**74.01**
RoBERTa$_{base}$	55.45	**58.73**
RoBERTa$_{large}$	**73.50**	70.62

across the models, except for RoBERTa$_{large}$, verifying that utilizing commonsense knowledge from external knowledge sources can help MNLMs infer commonsense knowledge.

V Conclusion

In this thesis, we empirically verified that questions that require semantic knowledge, especially commonsense knowledge, are still difficult for MNLMs while they show outstanding performance in various NLP tasks. Moreover, we suggested a possible solution to integrate semantic knowledge from external sources into the models. Our experimental results show that MNLMs can be complemented by injecting external commonsense knowledge. The analysis on the behavior of knowledge integration models remains as future work.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[2] T. Winograd, "Understanding natural language," *Cognitive psychology*, vol. 3, no. 1, pp. 1–191, 1972.

[3] M. Richardson, C. J. Burges, and E. Renshaw, "Mctest: A challenge dataset for the open-domain machine comprehension of text," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 193–203.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[5] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2019.

[6] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[7] T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3428–3448.

[8] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of NLP models with CheckList," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 4902–4912. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.442

[9] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[10] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 3319–3328.

[11] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[12] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International Conference on Machine Learning*, 2017, pp. 1885–1894.

[13] X. Han, B. C. Wallace, and Y. Tsvetkov, "Explaining black box predictions and unveiling data artifacts through influence functions," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5553–5563. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.492

[14] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni, "What you can cram into a single &!#* vector: Probing sentence embeddings for linguistic properties," in *Proceedings of Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2018, pp. 2126–2136.

[15] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 1073–1094.

[16] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. V. Durme, S. Bowman, D. Das, and E. Pavlick, "What do you learn from context? probing for sentence structure in contextualized word representations," in *Proceedings of the International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SJzSgnRcKX

[17] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4593–4601.

[18] J. Hewitt and C. D. Manning, "A structural probe for finding syntax in word representations," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4129–4138.

[19] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does bert look at? an analysis of bert's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019, pp. 276–286.

[20] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky, "Revealing the dark secrets of bert," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 4365–4374.

[21] S. Jain and B. C. Wallace, "Attention is not explanation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 3543–3556.

[22] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, 2019, pp. 2463–2473.

[23] N. Kassner and H. Schütze, "Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7811–7818.

[24] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: an open multilingual graph of general knowledge," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017, pp. 4444–4451.

[25] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[28] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[29] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of NAACL-HLT*, 2018, pp. 2227–2237.

[30] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International Conference on Machine Learning*, 2017, pp. 1243–1252.

[31] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[32] Ł. Kaiser and S. Bengio, "Can active memory replace attention?" in *Advances in Neural Information Processing Systems*, 2016, pp. 3781–3789.

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[34] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[35] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[36] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1400–1409.

[37] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 5149–5152.

[38] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 19–27.

[39] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1715–1725.

[40] S. Nagel, "Cc-news," http://web.archive.org/save/http://commoncrawl.org/2016/10/newsdataset-available, 2016.

[41] A. Gokaslan and V. Cohen, "Openwebtext corpus," http://web.archive.org/save/https://skylion007.github.io/OpenWebTextCorpus/, 2016.

[42] T. H. Trinh and Q. V. Le, "A simple method for commonsense reasoning," *arXiv preprint arXiv:1806.02847*, 2018.

[43] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu, "Open mind common sense: Knowledge acquisition from the general public," in *Proceedings of the International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*, 2002, pp. 1223–1237.

[44] L. Wang, M. Sun, W. Zhao, K. Shen, and J. Liu, "Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension," in *Proceedings of the International Workshop on Semantic Evaluation*, 2018, pp. 758–762.

[45] J. Guan, Y. Wang, and M. Huang, "Story ending generation with incremental encoding and commonsense knowledge," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6473–6480.

[46] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "Commonsenseqa: A question answering challenge targeting commonsense knowledge," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4149–4158.

[47] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 784–789.

[48] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[49] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

Acknowledgements

I thank those who have supported me and my research. I thank my advisor Jaesik Choi for guiding my research. I also thank my committee members, Kwang-In Kim and Sungbin Lim, for giving helpful comments. I would also like to appreciate every single member of our lab, SAIL. I thank my family for being always on my side and supporting everything for no purpose. Finally, I would like to say thank you once again to everyone who spends time helping and cheering me on.